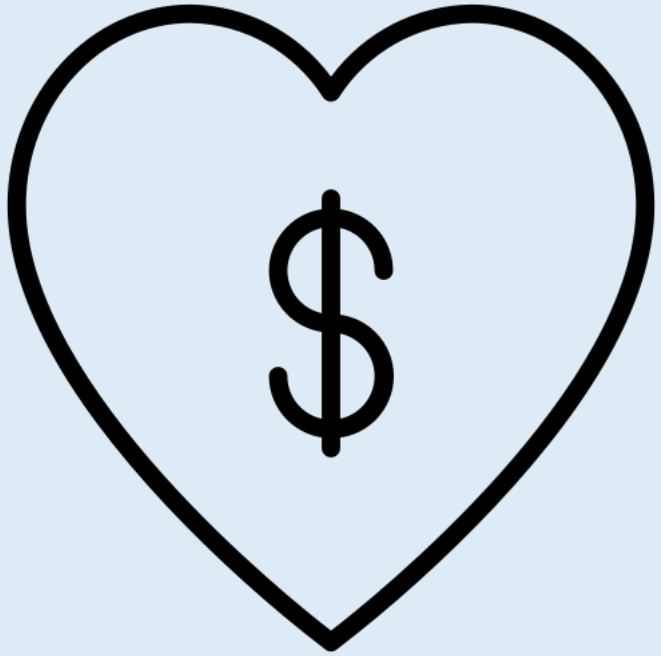
블록체인 기반 기부 플랫폼

Cointribute



2021-1 블록체인의 실무응용 1

A조 강명오, 유태윤, 이혜민, 정은선

Topic & Team

Topic

블록체인을 활용한 투명한 기부 플랫폼(dApp) 설계

Team

강명오 – 컨트랙트 개발 및 수정 / 발표 자료 develop

유태윤 – 발표 / 발표 자료 제작 / 컨트랙트 개발 및 수정

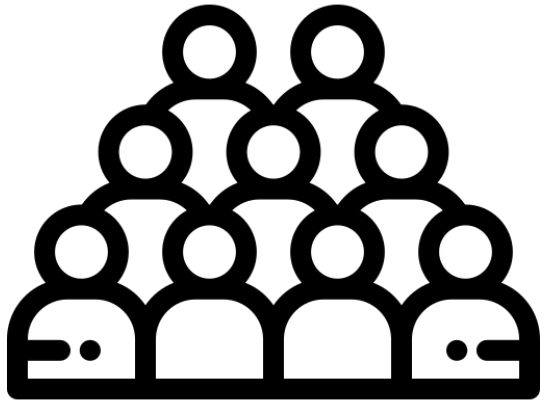
이혜민 – dApp 프로토타입 개발 / 주제 창안

정은선 – 컨트랙트 개발 및 수정 / 주제 창안

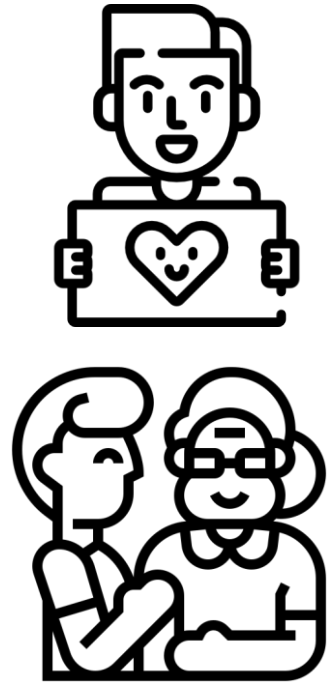
Design

플랫폼 형태의 dApp

- 두 종류의 User : 기부자(Donator) / 수혜자(Beneficiary)



Donator

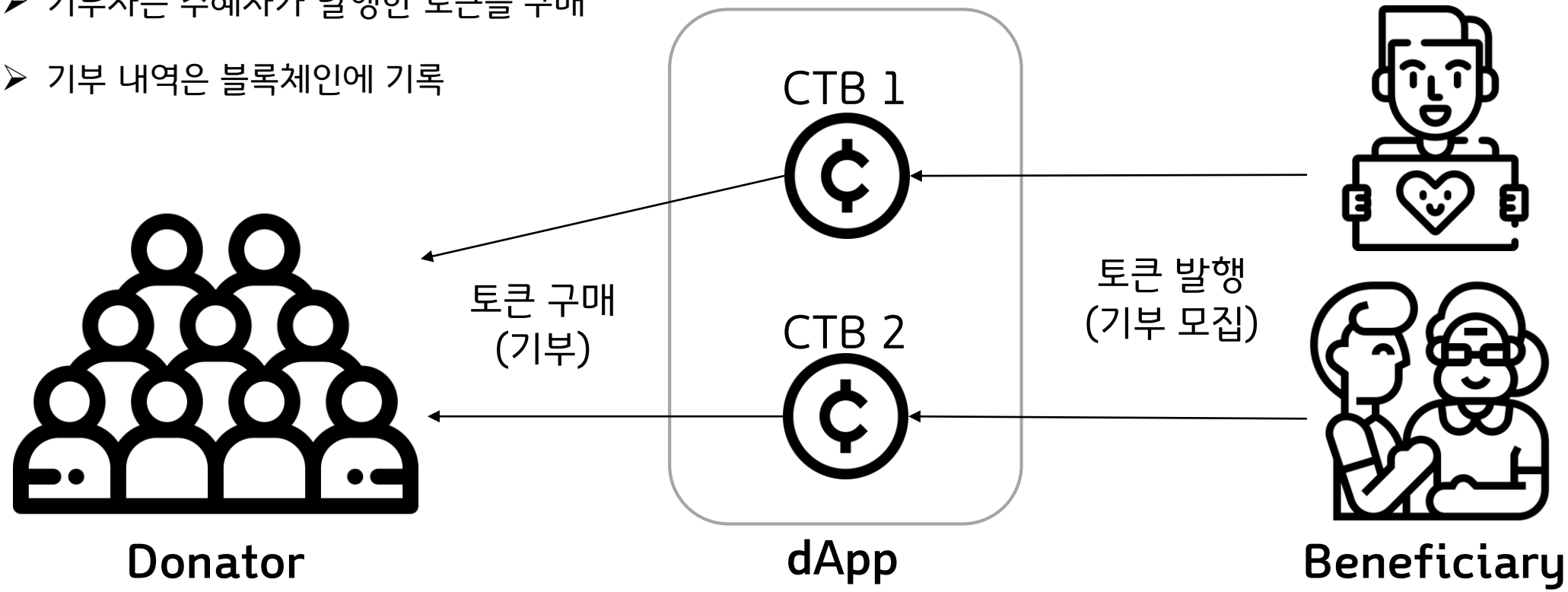


Beneficiary

Design

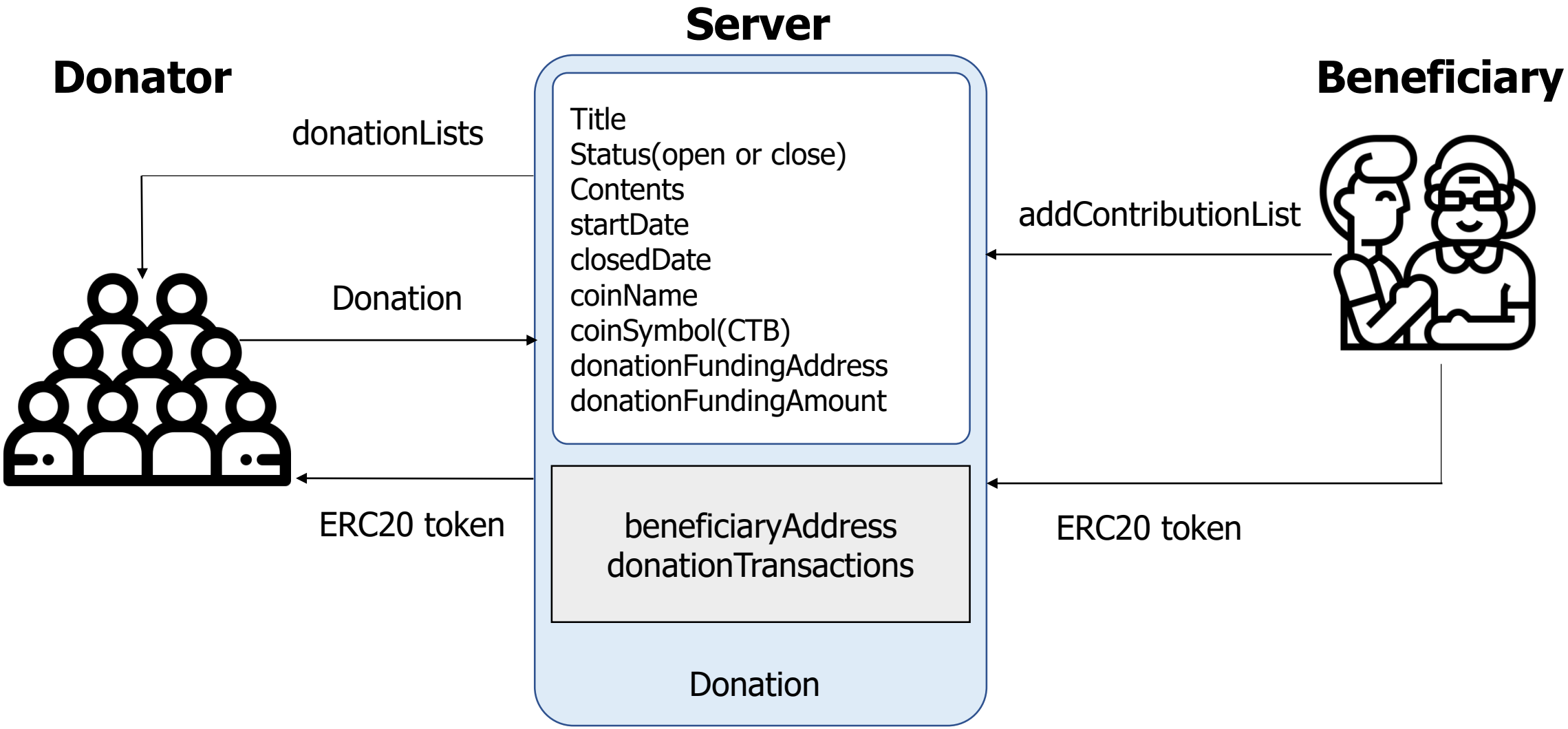
탈중앙화 방식의 투명한 기부

- 수혜자가 자체적으로 ERC20 기반 토큰(Cointribute – CTB) 발행
- 기부자는 수혜자가 발행한 토큰을 구매
- 기부 내역은 블록체인에 기록



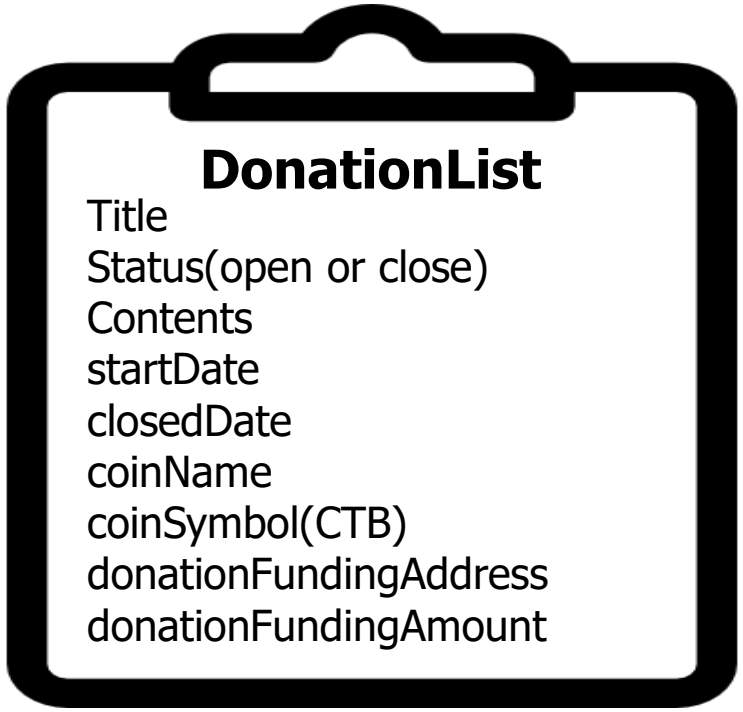
Contract Summary

Full Code : https://github.com/rkdauddh/BlockChain.git (./project/projectfinal.sol)



Contract

- 수혜자가 dApp에서 ERC20 토큰 발행 시 기부목록 등재
- 각 기부목록은 struct(DonationList)에 저장



- 각 Donation은 리스트(donationLists) 형식으로 저장
- 리스트의 index를 통해 각 Donation의 수혜자 주소 및 기부자 목록을 관리

Contract Detail

- Donation structure

```
// (구조체1) 기부리스트
struct DonationList {
    string title;                // 기부제목
    uint status;                // 기부상태 1:open, 2:close - 10이면 open, 2면 close
    string contents;             // 기부모집내용
    uint256 startDate;           // 기부시작일
    uint256 closedDate;          // 기부종료일
    string coinName;
    string coinSymbol;
    address donationFundingAddress; // 기부모집계좌
    uint256 donationFundingAmount; // 기부모집금액

}
```

```
    DonationList[] public donationLists;
```

Contract Detail

- Add contribution to Donation list

```
// (function 1) 수혜자가 화면에서 기부목록 등록시 호출한다.
// 1. _donationFundingAmount만큼 ERC20을 생성한다.
// 2. donationLists에 입력받은 기부목록정보를 insert한다.
// 3.기부된 금액 초기화
// 4.기부 참여자수 초기화
function addContributionList(string memory _title,
                             string memory _contents,
                             uint256 _startDate,
                             uint256 _term,
                             string memory _coinName,
                             string memory _coinSymbol,
                             uint256 _donationFundingAmount) public {

    uint _status;
    if(_startDate>block.timestamp){
        _status =2;
    }
    else{
        _status =1;
    }

    bool result = _createCoinbutor(donationListIndex, _coinName, _coinSymbol, _donationFundingAmount, msg.sender);  // 1. _donationFundingAmount만큼 ERC20을 생성한다.

    require(result, "ERROR:cannot create token");
    _addContributionList( _title, _status, _contents, _startDate, _startDate+_term, _coinName, _coinSymbol, msg.sender, _donationFundingAmount); // 2. donationLists에 입력
    donatedBalance[msg.sender] = 0;// 3.기부된 금액 초기화
    donatedCount[msg.sender] = 0;  // 4.기부 참여자수 초기화

}
```

Contract Detail

- Add contribution to Donation list (Internal functions)

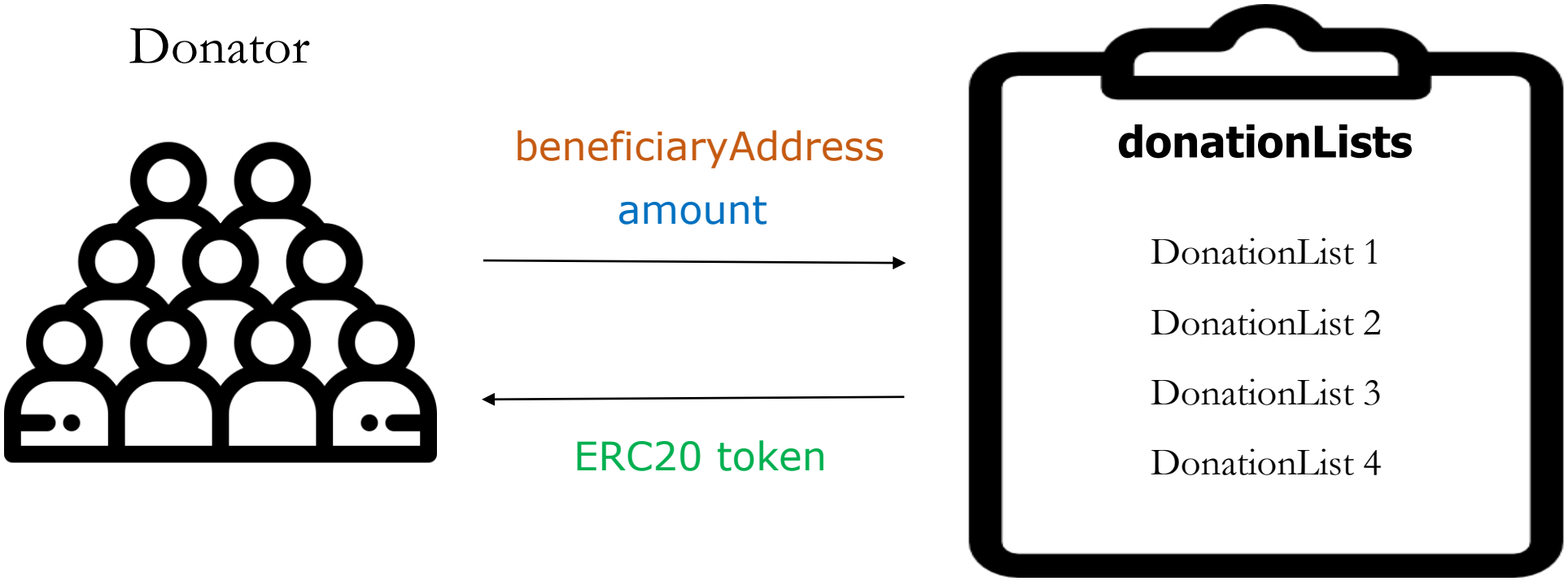
```
function _createCoinbutor(uint _index, string memory _name, string memory _symbol, uint256 _amount, address _donationFundingAddress) internal returns (bool) {
    _coinName[_index] = _name;
    _coinSymbol[_index] = _symbol;
    _totalSupply[_index] = _amount;
    _balances[_donationFundingAddress] = _amount;        //기부리스트 생성 초기에, 수혜자주소에 발행한 ERC20코인 총량을 넣는다. 이후 기부자들이 이를 구매시 차감된다.

    emit Transfer(address(0), _donationFundingAddress, _amount);
    return true;
}
```

```
    function _addContributionList(string memory _title,
                                  uint _status,
                                  string memory _contents,
                                  uint256 _startDate,
                                  uint256 _closedDate,
                                  string memory _coinName,
                                  string memory _coinSymbol,
                                  address _donationFundingAddress,
                                  uint256 _donationFundingAmount) internal {
        donationLists.push(DonationList( _title, _status, _contents, _startDate, _closedDate, _coinName, _coinSymbol, _donationFundingAddress, _donationFundingAmount));
        donationListIndexToBeneficary[donationListIndex] = _donationFundingAddress;
        beneficiaryToDonationListIndex[_donationFundingAddress] = donationListIndex++;
        donatedCount[_donationFundingAddress] = 0;
    }
```

Contract

- 기부자가 dApp에 들어오면 전체 기부리스트를 제공
- 기부자가 기부리스트 중에서 선택하여 기부금액을 입력
- 수혜자가 발행한 ERC20 token을 기부자가 구매



Contract Detail

- Get donation list for donator

```
// (function 2) 기부자가 기부화면에 들어오면 기부리스트를 가져와 보여준다.
// (requestType) 1: 전체리스트, 2: open상태인 리스트 전체, 3: close상태인 리스트 전체.. 로 하려다가 2번 구현시 에러나서 일단 스킵..;
function getDonationList(uint _requestType) public view returns (DonationList[] memory){
    uint idx=0;
    DonationList[] memory openDonationList;

    if(_requestType == 1){
        for(uint i=0; i<= donationListIndex; i++){
                openDonationList[idx++] = donationLists[i];
        }
    }
    else if(_requestType == 2){
        for(uint i=0; i<= donationListIndex; i++){
            if(donationLists[i].status == 1){
                openDonationList[idx++] = donationLists[i];
            }
        }
    }
        return openDonationList;
}
```

Contract Detail

- Donate

```
// (function 3) 기부자가 리스트 중 선택해서 기부금액 입력하고 submit버튼 클릭시 호출된다.
// 1. transfer() 함수를 통해 기부자는 수혜자가 발행한 ERC20 coin을 구매한다.

function Donation(address _beneficiaryAddr, uint256 _amount) public {
    uint idx = beneficiaryToDonationListIndex[_beneficiaryAddr];
    transfer(msg.sender, _beneficiaryAddr, _amount);
    _afterTransaction(idx, msg.sender, _beneficiaryAddr, _amount, block.timestamp);

}

function _afterTransaction(uint _donationListIndex, address _donatorAddress, address _beneficiaryAddr, uint256 _donatorAmount, uint256 _donationDate) internal {
    donationTransactions.push(DonationTransaction( _donationListIndex, _donatorAddress, _donatorAmount, _donationDate));  //donationTransactions List에 값을 push한다.
    donatedBalance[_beneficiaryAddr] = donatedBalance[_beneficiaryAddr] + _donatorAmount;
    donatedCount[_beneficiaryAddr]++;
}
```

Contract Detail

- Donate

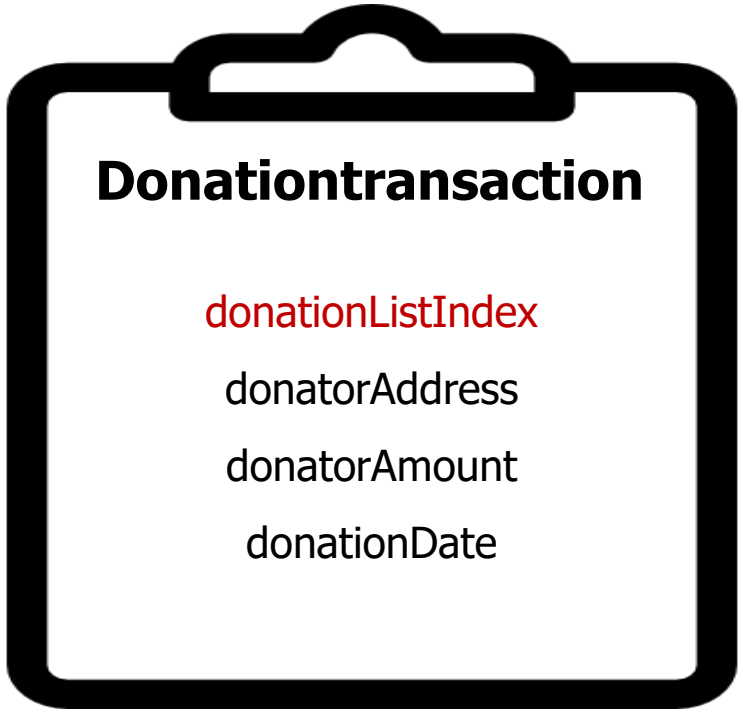
```
function transfer(address donatorAddr, address beneficiaryAddr, uint256 amount) public virtual returns (bool) {
    _transfer(donatorAddr, beneficiaryAddr, amount);
    return true;
}


// 기부자는 ETHER로 수혜자가 발행한 코인을 사는 방식으로 기부를 하는 형태임.
// 기부자의 ETHER잔액을 읽어와 차감하는 부분에 대한 구현은 생략함.
// 기부자가 수혜자의 ERC20 코인을 얻는 방식만 구현하였음
function _transfer( address _donatorAddr, address _beneficiaryAddr, uint256 _amount) internal virtual {
    require(_donatorAddr != address(0), "ERC20: transfer from the zero address");
    require(_beneficiaryAddr != address(0), "ERC20: transfer to the zero address");

    uint256 ableBalance = _balances[_beneficiaryAddr];
    require(ableBalance >= _amount, "ERC20: amount exceeds donation able amount");
    _balances[_beneficiaryAddr] = ableBalance - _amount;            // 수혜자의 잔액에서 기부금액만큼 차감 처리
    _balances[_donatorAddr] = _balances[_donatorAddr] + _amount;    // 기부자의 잔액에서 기부금액만큼 증감 처리
```

Contract

- 기부거래내역은 DonationTransaction struct에 저장
- 전체 기부거래내역은 DonationTransaction struct의 리스트로 저장



➢ Donation 리스트(donationLists)의 index 정보를 저장

Contract Detail

- DonationTransaction structure

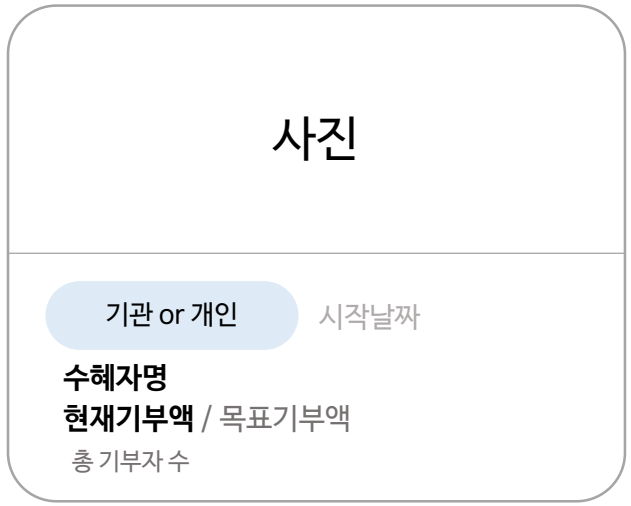
```
// (구조체2) 기부거래내역
struct DonationTransaction {
    uint donationListIndex; // 기부리스트의 인덱스
    address donatorAddress; // 기부자주소
    uint256 donatorAmount;  // 기부금액
    uint256 donationDate;     // 기부일자
}
```

```
DonationTransaction[] public donationTransactions;
```

dApp

Full code : https://github.com/hyemmie/cointribute-dapp

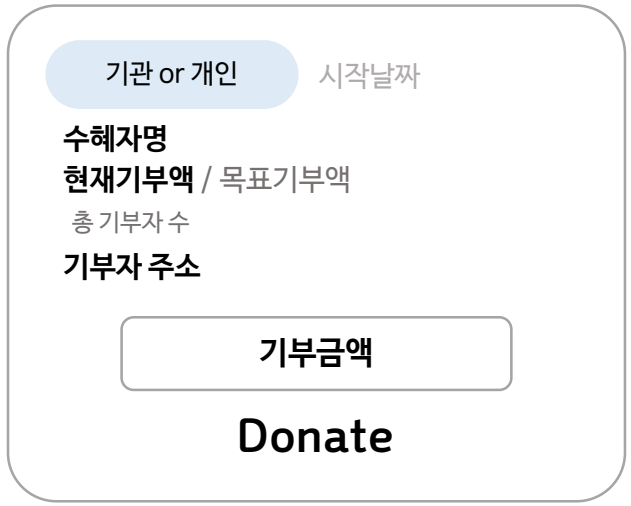
- **진행중인 기부목록 표시**



- **메타마스크 지갑에 연결하여 로그인하면, 현재까지의 기부내역 표시 (수혜자 / 최근 기부 일자 / 기부액)**

dApp

- 기부목록을 선택하여 기부



- 기부가 잘 진행되었는지 Transaction 결과 확인(Ehterscan)

Cointribute

Connect Wallet

Beneficiary List

Conclusion

가상화폐 구매에 대한 새로운 가치 제시

- ‘가상화폐 구매 = 투기’ 라는 부정적 시선 다수

Cointribute : ‘가상화폐 구매 = 기부’ 라는 새로운 가치 창출

기사제목출처 : https://www.yna.co.kr/view/AKR20210508048500009, https://www.hankyung.com/economy/article/202105292133g, https://www.hankookilbo.com/News/Read/A2021052713420005526

Thank you for listening!